

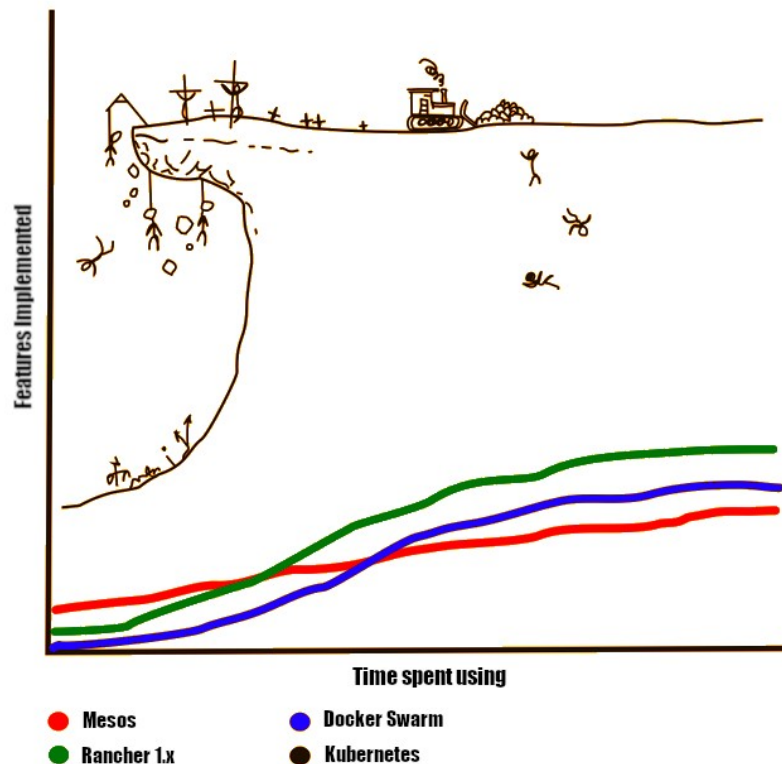


Introduction à Kubernetes

Rémi Cailletaud – CNRS/OSUG
MS ICD – IMT Atlantique

Introduction

Learning curves of some Container Orchestration Engines



La réputation d'un système complexe

...

est-elle méritée ?

Contexte : l'OSUG

- 8 unités de recherche, 5 équipes de recherche associées et 2 unités d'appui et de recherche.
 - 28 Service Nationaux d'Observation (SNO).
 - Soutien aux labo et aux SNO.
 - À la frontière des métiers ASR / Dev.
-

Contexte : les pratiques

Gestion de conf

- Peu de pratique communes.
- Trop de tâches d'administration pour les développeurs.

Les conteneurs

- De plus en plus de demandes.
 - Difficulté de gestion.
 - Question de sécurité.
-

Les conteneurs

- PyCon 2013... Tout juste 10 ans !
 - *Build once, run anywhere* : packaging de l'application et de ses dépendances. Runtime léger, magasin d'applications !
 - *Separation of Concerns* : le *Dev* est intéressé par l'intérieur du conteneur, l'*Ops* par l'extérieur (logging, monitoring, réseau...)
 - Normalisation du format (OCI Image Format) et des runtimes (OCI Runtime): *containerd*, *cri-o*, *rkt*, *kata*...
-

Le choix de Kubernetes

- Adoption par CoreOS et Rancher Labs.
 - Seul véritable concurrent, Docker Swarm.
 - Un choix motivé :
 - La modularité ;
 - L'intégration avec les infrastructures sous-jacentes ;
 - La gouvernance.
-

Kubernetes : historique

- Deux projets Google, dont Google Borg, en Java.
 - Réécriture en Go, version 1.0 en 2015.
 - Pilotage par Google, puis par la Cloud Native Foundation (Linux Foundation) depuis août 2018.
-



Kubernetes

«We must treat the datacenter itself as one massive warehouse-scale computer.»

in *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*

Luiz André

BarrosoJimmy Clidas

Urs Hölzle

Kubernetes : objectifs

- L'abstraction des couches matérielles et système.
- Le couplage faible des composants.
- Un surcoût minimal.
- Fonctionnement indifférent sur machines physiques et sur machine virtuelles.

L'OS du cloud

Kubernetes : concepts

Une API déclarative

On définit des «contrats» pour nos applications.

Kubernetes : concepts

Des capacités d'autoréparation



Jonathan Schaeffer 9 h 41

Salut Rémi !

Quand tu pourras, redémarre influxdb un petit coup, j'ai fait une trop grosse requête !



Jonathan Schaeffer 10 h 07

rémi, je crois qu'il faut encore rebooter influxdb



Rémi Cailletaud 10 h 07

héhé



Rémi Cailletaud 10 h 10

Lance-moi une instance d'InfluxDB.



Kubernetes 10 h 10

OK, je m'en occupe.



Kubernetes 10 h 25

Tiens, Jonathan a encore lancé une monstro-requête. Je redémarre l'instance.

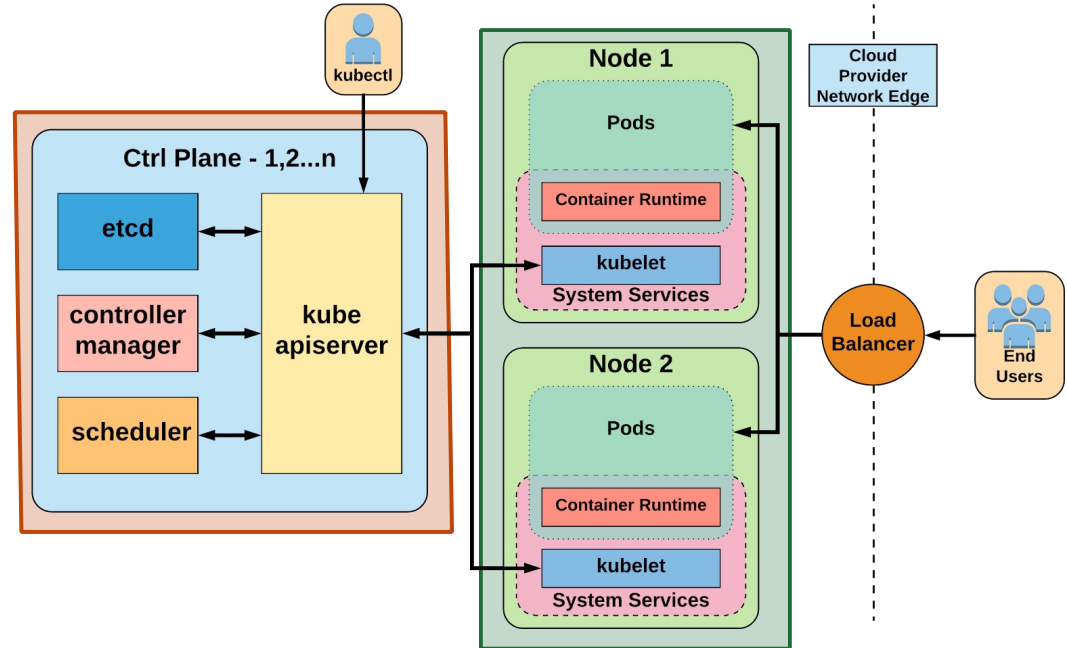
Kubernetes : concepts

Une infrastructure immutable

- Tests facilités.
 - Passage à l'échelle.
 - Mise à jour et retour en arrière.
-

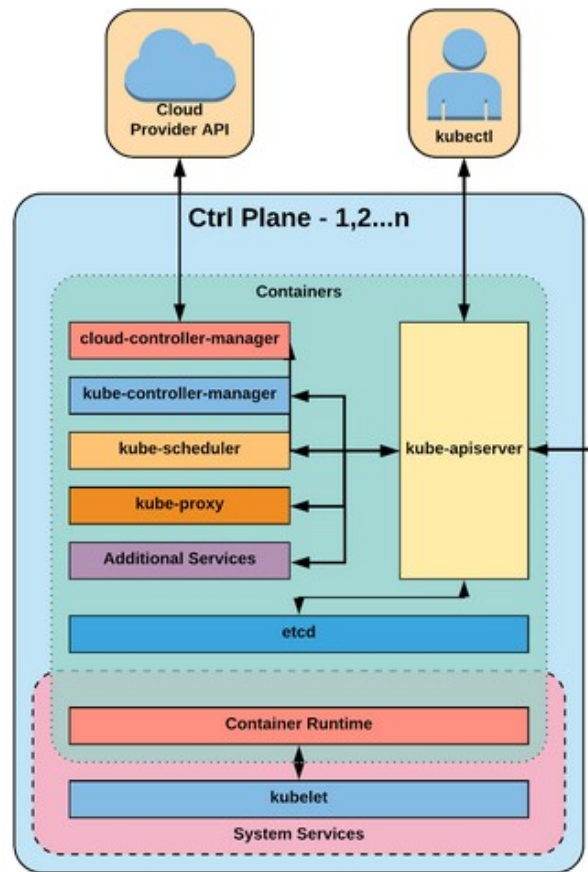
Kubernetes : architecture

- Une séparation nette du **plan de contrôle** et du **plan de travail**.
- Configuration par un point unique, via l'API.
- Composants faiblement couplés : communication uniquement avec l'API.



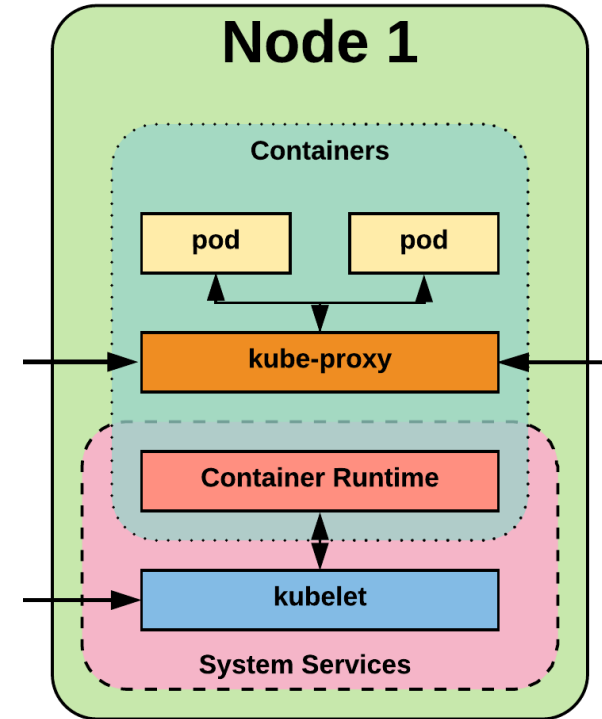
Kubernetes : le plan de contrôle

- L'API au centre du système.
- *etcd* pour le stockage clé/valeur.
- Des boucles de contrôles : *kube-controller-manager*.
- Un scheduler : *kube-scheduler*.
- L'intégration au cloud sous-jacent : *cloud-controller-manager*.



Kubernetes : le plan de travail

- Gestion des charges de travail (*pods*) : *kubelet*.
- Gestion des règles de *forwarding* et de l'équilibrage de charge : *kube-proxy*.
- Un *runtime* compatible *Container Runtime Interface* : Docker, containerd, rkt, CRI-O, Kata...



Les objets de base : Namespace

La version de l'API

Le type de l'objet

Le nom de l'objet

```
apiVersion: v1
kind: Namespace
Metadata:
  name: jres19
```

```
* k3d-jres19:default ~ kubect1 get namespaces
NAME          STATUS   AGE
default       Active   10m
kube-system   Active   10m
kube-public    Active   10m
kube-node-lease Active   10m
*k3d-jres19:default ~ kubect1 create namespace jres19
namespace/jres19 created
*k3d-jres19:default ~ kubect1 get namespaces   lun. 28 oct. 2019 21:30:51 CET
NAME          STATUS   AGE
default       Active   10m
kube-system   Active   10m
kube-public    Active   10m
kube-node-lease Active   10m
jres19        Active   6s
*k3d-jres19:default ~                               3490ms < lun. 28 oct. 2019 21:30:57 CET
```


Les objets de base : Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
spec:
  containers:
  - name: nginx
    image: nginx:1.17
    ports:
    - containerPort: 80
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
```

Ressource éphémère :
Nom et IP non persistants !

L'image utilisée.

Les ports exposés en interne.

Les ressources demandées et limites.

Les objets de base : Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jres19
spec:
  replicas: 3
  [ ... ]
  template:
    metadata:
      labels:
        app: jres19
        role: frontend
    spec:
      containers:
        - image: whoami
          name: whoami
          ports:
            - containerPort: 80
```

Gérés par le **Deployment**

```
* k3d-jres19:jres19 ~ kubectl get pods mer. 30 oct. 2019 15:15:58 CET
No resources found.
* k3d-jres19:jres19 ~ kubectl apply --recursive -f deploy.yml
deployment.apps/jres19 created
* k3d-jres19:jres19 ~ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
jres19-5b468b5f46-z9ntk            1/1     Running   0           4s
jres19-5b468b5f46-jwlbp            1/1     Running   0           4s
jres19-5b468b5f46-pnbh2            1/1     Running   0           4s
* k3d-jres19:jres19 ~ kubectl delete pod jres19-5b468b5f46-z9ntk
pod "jres19-5b468b5f46-z9ntk" deleted
* k3d-jres19:jres19 ~ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
jres19-5b468b5f46-jwlbp            1/1     Running   0           25s
jres19-5b468b5f46-pnbh2            1/1     Running   0           25s
jres19-5b468b5f46-fn7vd            1/1     Running   0           13s
* k3d-jres19:jres19 ~ mer. 30 oct. 2019 15:16:32 CET
```

Un template de pod.

Le nombre de réplicas.

Des labels...

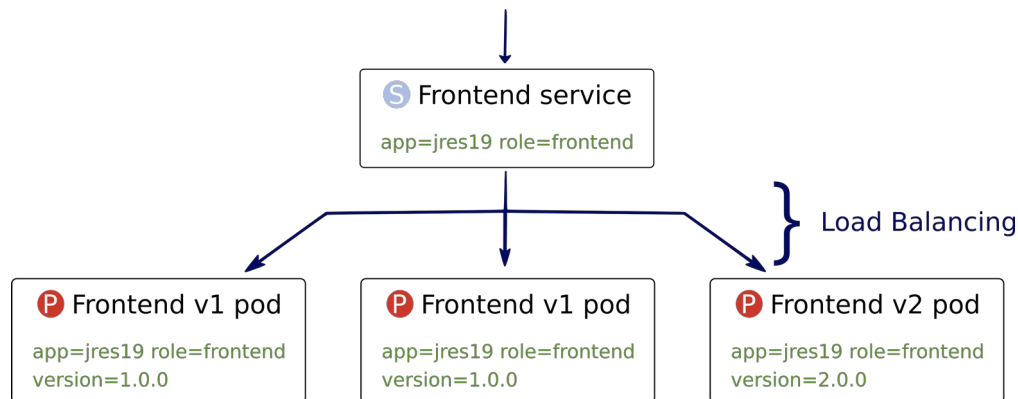
Les objets de base : ordonnancement

- **StatefulSet**: réseau, stockage et nom d'hôte persistants. Utile pour les applications statefull (et/ou en cluster).
 - **DaemonSet** : un pod par nœud. Utile pour le monitoring, les logs, le stockage...
-

Les objets de base : Service

- Ressource durable qui permet d'exposer les pods.
- Adresse IP et nom DNS statiques : **jres-svc.jres19.svc.cluster.local**

```
apiVersion: v1
kind: Service
metadata:
  name: jres19-svc
  Namespace:
spec:
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: jres19
    role: frontend
```



Sélection des pods de destination par label

Les objets de base...

- Job, CronJob : gestion des tâches.
 - Volume, PersistentVolume, PersistentVolumeClaim : gestion des volumes persistants.
 - ConfigMap, Secret : gestion de la configuration et des secrets.
 - Ingress : gestion du reverse-proxy.
-

Custom Resources Definition et Operators

- CRD : extension de l'API Kubernetes, de nouveaux objets
 - Custom Controllers
 - Operator Pattern : automatisation des processus de déploiements en utilisant CRDs et Custom Controllers
 - Exemples: ElasticSearch, PostgreSQL, Prometheus, ...
 - Voir <https://operatorhub.io/>
-

Helm: *The package manager for Kubernetes*

- Gestion de *packages* (*charts*) pour Kubernetes.
 - En fait, un moteur de templating avec la possibilité de publier les charts.
 - Devenu le standard pour la **distribution** d'application.
 - Des alternatives existent : kustomize, jsonnet, ksonnet.
-

Argo CD: *Declarative GitOps CD for K8s*

- GitOps: gestion de l'infrastructure et des configurations d'applications qui reposent sur l'utilisation de Git.
 - Une unique source de vérité pour la formalisation déclarative de l'infrastructure et des applications.
 - Fonctionne avec kustomize, Helm, jsonnet, ksonnet.
 - Intégration d'outils tiers facile.
 - Bootstrap de cluster !
-

CNCF: *Cloud Native Computing Foundation*

- Projet de la Linux Foundation.
 - Lancé en 2015 pour aider à faire progresser les technologies conteneurs.
 - 450 membres, dont Google, Red Hat, Twitter, Huawei, Intel, Cisco, IBM, Docker, et VMware.
 - Pilotage de Kubernetes depuis 2018.
 - 16 projets *Gratuated*, 26 projets *Incubating*, 69 projets *Sandbox* .
-



CNCF: *Cloud Native Computing Foundation*



Conclusion

Fiable,
robuste,
complet,
extensible,
pas si complexe.

De nouveaux
paradigmes, un
ecosystème très
(très(très))
dynamique...



